Local Response Surface Approximation in Evolutionary Algorithms for Optimization of Costly Functions

Rommel G. Regis, Christine A. Shoemaker, *Member, IEEE*

R. G. Regis is with the School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853. E-mail: rregis@orie.cornell.edu .

C. A. Shoemaker is Ripley Professor of Engineering, School of Civil and Environmental Engineering, Cornell University, Ithaca, NY 14853. E-mail: cas12@cornell.edu .

Abstract

We develop an approach for the optimization of continuous costly black box functions that uses space-filling experimental designs and local response surface approximations to reduce the number of function evaluations in an evolutionary algorithm. Our approach is to estimate the objective function value of an offspring solution by fitting a response surface model over the k nearest previously evaluated points, where $k = (d+1)(d+2)/2$ and d is the dimension of the problem. The estimated function values are used to screen offspring to identify the most promising ones for the costly function evaluation. To fit response surface models, a space-filling experimental design is used to determine initial points for costly function evaluation. We compared the performance of a (μ, λ)-ES with local quadratic approximation, a (μ, λ)-ES with local radial basis function (RBF) interpolation, and a conventional (μ, λ)-ES which has no local approximation. The experimental design used was a symmetric Latin hypercube and the RBF interpolant has a cubic form augmented by a linear polynomial. The performance of these algorithms were compared on the Dixon-Szegö test functions and on the 10-dimensional Rastrigin and Ackley test functions. All comparisons involve multiple trials, ANOVA analyses, and computation of simultaneous confidence intervals to determine if the observed differences in performance are statistically significant. The results indicate that (μ, λ)-ES algorithms with local approximation were significantly better than conventional (μ, λ)-ES algorithms on the Dixon-Szegö test functions. However, for the more difficult 10-dimensional Rastrigin and Ackley test functions, only the RBF approach was successful in improving the performance of a (μ, λ)-ES. Moreover, the results also suggest that the RBF approach is superior to the quadratic approximation approach on all test functions although the difference in performance is statistically significant only for the harder 10-dimensional test functions.

Keywords

Optimization, Costly Function, Evolutionary Algorithm, Response Surface, Quadratic Regression, Radial Basis Function, Latin Hypercube.

I. Introduction

A. Problem Definition and Motivation

Evolutionary algorithms initially focused on solving combinatorial problems or on continuous optimization problems for which objective function evaluation is fast. However, global optimization problems also arise for continuous functions whose evaluation is computationally expensive (e.g. minutes to hours for each function evaluation). For such functions, an analyst is typically willing to perform only a small number of function evaluations to solve the optimization. Conventional evolutionary algorithms often cannot find good solutions with a limited number of iterations. Our goal in this paper is to enhance evolutionary computation by using local response surface approximations to screen out less suitable offspring and hence to limit function evaluations to offspring that appear most promising. Success of this approach improves one's ability to find good solutions for optimization problems of costly nonconvex functions since the additional time required to compute response surface approximations is small in comparison to the time required for objective function evaluation.

We now provide a precise statement of the problem we wish to solve. Let $\mathcal{D} \subseteq R^d$ be a compact set and let $f : \mathcal{D} \to R$ be a (deterministic) continuous function. The *global optimization problem* is to find $x^* \in \mathcal{D}$ such that $f(x^*) \leq f(x)\ \forall x \in \mathcal{D}$. Note that under the given conditions, f attains its global minimum value on $\mathcal{D}$. In this paper, we would like to focus on global optimization problems where f is a black box function

that is costly to evaluate. For simplicity, we assume that the domain $\mathcal{D}$ is a box in R^d, i.e.

$$\mathcal{D} = \{x \in R^d : -\infty < a_i \leq x_i \leq b_i < \infty,\ i = 1, \ldots, d\}, \tag{1}$$

for some $a_i, b_i \in R$. Furthermore, we also assume that the derivatives of f are too expensive to accurately compute so that they are practically unavailable. In other words, the only thing we have about f is the ability to evaluate it at any point on its domain. Since f is costly to evaluate, the goal is to find an approximate global minimizer for f on $\mathcal{D}$ using as few function evaluations as possible.

Optimization of costly continuous black box functions has enormous potential in engineering where models describing the system often require lengthy simulation of complex computer codes. Optimization for these kinds of problems is important both in system design to maximize performance and in parameter estimation (the inverse problem). One important class of especially costly functions are those that require solutions of systems of partial differential equations, where nonlinearities and accuracy requirements necessitate small time steps and fine meshes, resulting in large computation times for each model simulation.

B. Literature Review

There are shortcomings with most of the existing methods for solving optimization problems for costly black box functions. Gradient-based algorithms cannot be used in many cases simply because accurate derivatives are not available. Evolutionary algorithms and other modern heuristics like simulated annealing typically require a very large number of function evaluations to obtain adequately good solutions for higher dimensional problems.

An alternative to gradient-based and heuristic methods for optimizing computationally expensive functions are methods that are based on response surface models (also known as metamodels). The most popular of these methods is traditional response surface methodology ([1], [2], [3]) which generally involves low-order polynomial regression. Other response surface optimization methods are those that rely on kriging models ([4], [5], [6], [7]), radial basis functions ([8], [9], [10], [11], [12], [13], [14]), and neural networks ([15], [16], [17], [18], [19]). These procedures operate by maintaining an approximation of the underlying function to be optimized. The approximate model is used to identify promising points for function evaluation. Note that the global minimum in the approximate model does not always correspond to a global minimum of the actual surface. Hence, some of these methods tend to be iterative in the sense that the approximating surface is periodically refitted upon the addition of newly evaluated points. However, a naive implementation of these methods, where the global minimizer of the current approximating surface is always selected for function evaluation may converge to some point which may not even be a local minimizer of the actual function ([4], [9]). Moreover, many global response surface methods have difficulty fitting bumpy surfaces based on a limited number of function evaluations and may focus the search on the area near the current best solution rather than identifying a promising area for future search.

Another approach to optimizing costly black box functions is to use response surface models to speed up evolutionary algorithms. The basic idea in this approach is to maintain an approximate model of the

underlying function and use it to estimate the function values of the offspring in any generation. Instead of evaluating the costly function at each offspring in a given generation, we simply evaluate it at a subset of the offspring solutions (i.e. the offspring solutions with the lowest estimated function value). With this approach, the evolutionary algorithm can continue for more generations than it could with a standard implementation given a fixed limit on the number of costly function evaluations. By doing this, we are more likely to obtain better quality solutions for a fixed number of function evaluations than with the standard implementation of evolutionary algorithms.

Various response surface models have been used to approximate fitness functions in evolutionary computation. For instance, Ratle ([20], [21]) and El-Beltagy et al. [22] used kriging interpolation, El-Beltagy and Keane [23] used Gaussian processes while Jin et al. ([24], [25], [26]) used neural networks to approximate fitness functions. Rasheed [27] approximated fitness functions in genetic algorithms by clustering the points encountered during the optimization and by periodically forming quadratic approximations over the entire set of evaluated points as well as over large enough clusters. A more recent paper by Rasheed et al. [28] compared the performance of quadratic regression, radial basis function networks, and quickprop neural networks in speeding up genetic-algorithm-based design optimization. In that paper, the authors found that quadratic regression was the best among the three metamodeling techniques on some engineering design problems.

Jin et al. ([24], [25], [26], [29]) coined the term *evolution control* to refer to the process of using the actual objective function from time to time when evaluating the fitness of offspring solutions. They pointed out that evolution control is needed in order to address the problem of incorrect convergence of an evolutionary algorithm in the presence of false minima in the approximate model. Jin et al. ([25], [26]) also introduced a framework for evolutionary optimization with approximate fitness functions where the frequency of evolution control is based on an estimate of the local fidelity of the approximation model. Finally, a recent survey paper by Jin [29] outlined several approximation models and data sampling techniques for use with evolutionary computation.

C. Proposed Method

In this paper, we propose an approach for costly black box optimization that uses space-filling experimental designs and k-nearest neighbor local response surface approximations to improve the performance of an evolutionary algorithm. The local approximation is used to assess the wisdom of doing an actual function evaluation for each offspring of an evolutionary algorithm so that the number of costly function evaluations can be reduced, thereby improving computational efficiency. By k-nearest neighbor local approximation, we mean that the objective function value (or the fitness value) of an offspring solution will be estimated by fitting a model using its k-nearest neighbors among the previously evaluated points. Here, k is set equal to $(d+1)(d+2)/2$ which is the minimum number of data points required to fit a quadratic model. The justification for setting k to this value is that the quadratic model is one of the simplest smooth surfaces that can capture nonlinearity. In the context of engineering design optimization using genetic algorithms, Rasheed

[30] used k-nearest neighbors to classify design points as feasible, infeasible-evaluable, and unevaluable. El-Beltagy et al. [22] suggested the use of a local metamodel in the context of kriging interpolation in order to alleviate the computational burden of fitting metamodels. To the best of our knowledge, none of the previous work in evolutionary computation used local response surface approximations via k-nearest neighbors to approximate objective function values of offspring solutions.

In the numerical experiments, two response surface models will be used for local approximations of the costly function: quadratic regression and radial basis function (RBF) interpolation. The RBF interpolation model that will be used here is based on the work done by Powell ([13], [14]) of the Cambridge Numerical Analysis Group and is more general than a typical RBF network model since it is equivalent to an RBF network model augmented by a low-order polynomial. Moreover, a typical RBF network uses Gaussian RBFs whereas in this investigation, we will use a cubic RBF with a linear polynomial tail. Recent studies suggest that the cubic and thinplate RBFs have more desirable theoretical properties than Gaussian or multiquadric RBFs [31]. A space-filling experimental design ([32], [33]) provides an excellent set of points where the initial function evaluations should take place in order to get information needed to fit a response surface model. The function values at the experimental design points provide a rough global picture of the underlying function. In this investigation, we will use a symmetric Latin hypercube design [34]. Finally, the particular evolutionary algorithm that will be used in this investigation is a (μ, λ)-Evolution Strategy with uncorrelated mutations which was originally proposed by Schwefel [35]. However, our method can be used with most evolutionary algorithms.

In the computational experiments, the algorithms will be tested on some benchmark test functions for global optimization such as the Dixon-Szegö test functions [36], the 10-dimensional Rastrigin test function [37], and the 10-dimensional Ackley test function [38]. In addition, we will employ standard statistical techniques such as ANOVA with simultaneous confidence intervals to compare the performance of a (μ, λ)-ES with its enhanced versions. The purpose of the statistical analysis is to provide stronger and more solid claims regarding the relative performance of the different algorithms being tested.

This paper differs from those reviewed above in several ways. First, as pointed out earlier, this is the first paper to implement local response surface approximation via k-nearest neighbors in the context of evolutionary computation. Second, it is the only paper to use a radial basis function (RBF) augmented by a polynomial and also the only paper to use a cubic RBF (instead of the Gaussian form which is more popular in the machine learning community) with an evolutionary algorithm. Moreover, this paper is the first to use space-filling experimental designs to select initial points for function evaluation which are needed to fit a response surface model in the context of evolutionary computation. Finally, it is also the only paper on using response surface approximation in evolutionary algorithms to compute simultaneous confidence intervals on the results on a number of benchmark test functions for alternative algorithms. As we will show later, the combination of a symmetric Latin hypercube design and the cubic RBF with a linear polynomial tail generate a very effective local response surface approximation that enhances the performance of an evolution strategy.

II. Evolutionary Algorithms

A. Overview

Figure 1 outlines a typical evolutionary algorithm ([39]). $P(t)$ denotes the population at generation t, $Q(t)$ is a special set of individuals that has to be considered for selection during generation t (i.e. $Q(t) = P(t)$ for an elitist algorithm, or $Q(t) = \emptyset$ for a non-elitist algorithm), $P''(t)$ represents the offspring population that has been generated by means of mutation and/or recombination. Populations are evaluated by computing the objective function values (or sometimes fitness values) and selection is made by choosing the individuals with the best objective function values.

There are three main types of evolutionary algorithms [39], [40], [41]: *genetic algorithms (GAs)* ([42], [43]), *evolution strategies (ESs)* ([35], [44], [45], [46], [47], [48], [49]), and *evolutionary programming (EPs)* ([50], [51]). Genetic algorithms have been used extensively for solving mostly discrete optimization problems. On the other hand, evolution strategies and evolutionary programming algorithms have been used mostly on continuous optimization problems. In the context of optimization in real-valued search spaces, one advantage of ESs and EPs over ordinary GAs is that they allow self-adaptation of the parameters of the algorithm. In an ES or an EP, typical parameters are the standard deviations of the normal random mutation on the different components of a solution. By self-adaptation of parameters, we mean that the algorithm not only evolves a good set of solutions but it also evolves a good set of parameter values.

In this investigation, we use a particular type of evolution strategy, called (μ, λ)-ES, which was originally proposed by Schwefel [35]. Our choice of this particular evolutionary algorithm does not imply that we think this is the best evolutionary algorithm to use for continuous global optimization. We selected this algorithm because this is among the simplest of the evolutionary optimization algorithms that perform reasonably well on continuous optimization problems. Note that our goal is simply to demonstrate how we can use local response surface approximations to enhance an evolutionary algorithm and the (μ, λ)-ES appears to be a sufficient example for this purpose. We shall describe this algorithm in detail in the next section.

B. Description of the (μ, λ)-ES with Uncorrelated Mutations

Suppose we wish to minimize a real-valued function f of d continuous variables. We shall represent each solution as a vector x of length d. In a typical ES, an individual is represented as a vector (x, σ), where $x \in R^d$ is a feasible solution to the problem and $\sigma \in (R^+)^d$ is the parameter vector (i.e. vector of standard deviations of the normal random mutations on the different components of a solution) that gave rise to x. The individuals in generation g will be denoted by (x^g, σ^g) and the j^{th} component of the vectors x^g and σ^g will be denoted by $x^g(j)$ and $\sigma^g(j)$, respectively. Below is a detailed description of the (μ, λ)-ES suggested by Schwefel [35] as described in Bäck [40]. In the algorithm below, μ is the number of parents and λ is the number of offspring in each generation.

(μ, λ)-ES with Uncorrelated Mutations

(1). Set $\tau' = 1/\sqrt{2d}$ and set $\tau = 1/\sqrt{2\sqrt{d}}$.

(2). Set the generation index $g = 0$. Generate the initial set of parents $(x_1^0, \sigma_1^0), \ldots, (x_\mu^0, \sigma_\mu^0)$.

(3). Set $g := g + 1$. Generate λ intermediate offspring $(u_1^g, \alpha_1^g), \ldots, (u_\lambda^g, \alpha_\lambda^g) \in R^d$ by discrete recombination on the solution variables and (panmictic) intermediate recombination of the mutation rates. That is, for each $i = 1, \ldots, \lambda$, we perform the following steps:

(a). Select two parents by selecting two distinct indices s_i, t_i uniformly at random from the set $\{1, \ldots, \mu\}$. Then for each $j = 1, \ldots, d$, generate a uniform random number $\omega_{i,j} \in [0, 1]$ and set

$$u_i^g(j) = \begin{cases} x_{s_i}^{(g-1)}(j) & \text{if } \omega_{i,j} < 1/2 \\ x_{t_i}^{(g-1)}(j) & \text{otherwise} \end{cases} \tag{2}$$

(b). Select an index h_i uniformly at random from the set $\{1, \ldots, \mu\}$. Then for each $j = 1, \ldots, d$, select another index $k_{i,j}$ uniformly at random from the set $\{1, \ldots, \mu\}$, and set

$$\alpha_i^g(j) = (\sigma_{h_i}^{(g-1)}(j) + \sigma_{k_{i,j}}^{(g-1)}(j))/2 \tag{3}$$

(4). Modify the λ intermediate offspring $(u_1^g, \alpha_1^g), \ldots, (u_\lambda^g, \alpha_\lambda^g)$ into $(v_1^g, \beta_1^g), \ldots, (v_\lambda^g, \beta_\lambda^g)$ by applying the following mutation operation. For each $i = 1, \ldots, \lambda$, we perform the following steps:

(a). Generate random numbers $\xi_i, z_{i,1}, \ldots, z_{i,d}, \zeta_i \sim N(0, 1)$.

(b). For each $j = 1, \ldots, d$, set

$$\beta_i^g(j) = \alpha_i^g(j) \cdot \exp(\tau' \xi_i + \tau z_{i,j}) \tag{4}$$

$$v_i^g(j) = u_i^g(j) + \beta_i^g(j) \cdot \zeta_i, \tag{5}$$

(5). Compute the function values $f(v_1^g), \ldots, f(v_\lambda^g)$ associated with the λ offspring $(v_1^g, \beta_1^g), \ldots, (v_\lambda^g, \beta_\lambda^g)$.

(6). Rank the offspring according to their associated function values. Select the μ offspring corresponding to the lowest function values. This will be the set of parents for the next generation, i.e. $(x_1^g, \sigma_1^g), \ldots, (x_\mu^g, \sigma_\mu^g)$ will be the individuals from $\{(v_1^g, \beta_1^g), \ldots, (v_\mu^g, \beta_\mu^g)\}$ with the lowest associated function values.

(7). If the stopping criteria are not satisfied, return to 2.

A few remarks are in order. First, in the standard implementation, the initial parent solutions are selected uniformly at random throughout the entire domain. Second, based on investigations with a particular objective function (the sphere model), Schwefel [52] provided some guidelines for selecting μ and λ. In particular, he suggested that μ should be larger than one and $\mu/\lambda \approx 1/7$ to optimize the accelerating effect of self-adaptation. Third, in the computational experiments, we found $\sigma_i^0 = 0.05 \min_{1 \le i \le d}(b_i - a_i)$, $i = 1, \ldots, d$ to be a reasonable setting for the initial mutation rates for the test functions considered. Finally, note that step 2 could result in an offspring that is outside the box domain. In this case, we simply replace x_i^g by $\max(a, \min(x_i^g, b))$, where max and min are performed componentwise.

III. Response Surface Approximation Models

As mentioned earlier, we will use local response surface approximation to improve the performance of an evolutionary optimization algorithm on costly black box functions. The local approximation will be

implemented as follows. Suppose we are optimizing a continuous function f defined on a box $\mathcal{D} \subseteq R^d$. Let $S \subseteq \mathcal{D}$ be the set of points where the function values are known and let $k = (d+1)(d+2)/2$. To estimate the function value at a point $u \in \mathcal{D}$, we use the information on the function values of the k nearest neighbors of u among the points in S to build a response surface model $\hat{f}_u$ that approximates f at the vicinity of u. An estimate of $f(u)$ is then given by $\hat{f}_u(u)$.

A. Polynomial Regression

There are several response surface models that can be used to obtain local approximations. One of the simplest is polynomial regression which we now discuss. For convenience, we first define Π_m^d to be the linear space of polynomials in d variables of degree less than or equal to m. We also define $\Pi_{-1}^d = \{0\}$. Assume that we are given n distinct points $x_1, \ldots, x_n \in R^d$ and that we know the function values $y_1 = f(x_1), \ldots, y_n = f(x_n)$ at these points. Now suppose we wish to fit a polynomial of degree m in d variables around a query point $u \in R^d$ using the data points $(x_1, y_1), \ldots, (x_n, y_n)$. Let $\hat{m}$ be the dimension of Π_m^d and let $p_1, \ldots, p_{\hat{m}}$ be the natural basis of this linear space. (By a simple combinatorial argument, it is easy to see that $\hat{m} = \binom{m+d}{d}$.) In this case, we wish to fit a function of the form

$$p(x) = \sum_{i=1}^{\hat{m}} c_i p_i(x), \quad x \in R^d \tag{6}$$

where $c_i \in R$ for $i = 1, \ldots, \hat{m}$. In polynomial regression, we determine the coefficients c_i that minimize the sum of squared residuals

$$\sum_{k=1}^{n} \left(y_k - \sum_{i=1}^{\hat{m}} c_i p_i(x_k) \right)^2 \tag{7}$$

Let $\hat{c} = (\hat{c}_1, \ldots, \hat{c}_{\hat{m}})$ be the regression coefficients that minimize the expression (7) and let

$$\hat{p}(x) = \sum_{i=1}^{\hat{m}} \hat{c}_i p_i(x), \quad x \in R^d \tag{8}$$

Moreover, define

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad P = \begin{pmatrix} p_1(x_1) & \ldots & p_{\hat{m}}(x_1) \\ \vdots & & \vdots \\ p_1(x_n) & \ldots & p_{\hat{m}}(x_n) \end{pmatrix}. \tag{9}$$

Assuming that P has full column rank, it follows from elementary statistical theory that

$$\hat{c} = (P^T P)^{-1} P^T y. \tag{10}$$

In the case of quadratic regression, $\hat{m} = (d+1)(d+2)/2$ and the goal is to fit a function of the form:

$$p(t) = \beta_0 + \sum_{i=1}^{d} \beta_i t_i + \sum_{i=1}^{d} \sum_{j=i}^{d} \beta_{i,j} t_i t_j, \ t = (t_1, \ldots, t_d) \in R^d \tag{11}$$

A good reason for using quadratic polynomials in the context of local approximation by nearest neigbors is that it is among the simplest surface that can capture nonlinearity.

B. Radial Basis Function Interpolation

An alternative to polynomial regression is to use a response surface model based on radial basis functions. As before, assume that we are given n distinct points $x_1, \ldots, x_n \in R^d$ where the function values are known. In this method, we use an interpolant of the form

$$s(x) = \sum_{i=1}^{n} w_i \phi(||x - x_i||_2) + p(x), \quad x \in R^d \tag{12}$$

where $w_i \in R$ for $i = 1, \ldots, n$, p is in Π_m^d (the space of polynomials in d variables of degree less than or equal to m), and ϕ is one of the following forms:

$$\left.\begin{array}{lcll} \phi(r) & = & r & \text{(linear)}, \\ \phi(r) & = & r^3 & \text{(cubic)}, \\ \phi(r) & = & r^2 \log r & \text{(thin plate spline)}, \\ \phi(r) & = & \sqrt{r^2 + \gamma^2} & \text{(multiquadric)}, \\ \phi(r) & = & e^{-\gamma r^2} & \text{(Gaussian)}, \end{array}\right\} \quad r \geq 0, \tag{13}$$

where γ is a positive constant.

Fix ϕ. Define the matrix $\Phi \in R^{n \times n}$ by:

$$(\Phi)_{ij} := \phi(||x_i - x_j||), \quad i, j = 1, \ldots, n. \tag{14}$$

Moreover, define

$$m_\phi = \begin{cases} -1 & \text{if } \phi \text{ is Gaussian} \\ 0 & \text{if } \phi \text{ is linear or multiquadric} \\ 1 & \text{if } \phi \text{ is cubic or the thin plate spline} \end{cases} \tag{15}$$

and let $m \geq m_\phi$. As before, let $\hat{m}$ be the dimension of the linear space Π_m^d, let $p_1, \ldots, p_{\hat{m}}$ be a basis of this linear space, and define the matrix P as in (9).

In this model, the RBF that interpolates the points $(x_1, f(x_1)), \ldots, (x_n, f(x_n))$ is obtained by solving the system

$$\begin{pmatrix} \Phi & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} w \\ c \end{pmatrix} = \begin{pmatrix} F \\ 0_{\hat{m}} \end{pmatrix}, \tag{16}$$

where $F = (f(x_1), \ldots, f(x_n))^T$, $w = (w_1, \ldots, w_n)^T \in R^n$ and $c = (c_1, \ldots, c_{\hat{m}})^T \in R^{\hat{m}}$. Powell [13] showed that the matrix

$$A = \begin{pmatrix} \Phi & P \\ P^T & 0 \end{pmatrix} \in R^{(n+\hat{m}) \times (n+\hat{m})} \tag{17}$$

is nonsingular if and only if $x_1, \ldots, x_n$ satisfy the property:

$$q \in \Pi_m^d \quad \text{and} \quad q(x_i) = 0, \ i = 1, \ldots, n, \quad \Rightarrow \quad q \equiv 0. \tag{18}$$

(In the Gaussian case with $m = -1$, P and the above condition are omitted.) Hence, in this case, the resulting RBF interpolant $s(x)$ is unique.

In the later numerical investigation, we used a particular RBF model where ϕ is cubic and $p(x)$ is a linear polynomial. There are theoretical and numerical reasons why we picked the cubic form. Recent studies by Gutmann [31] indicate that the linear, cubic, and thinplate RBFs have better theoretical properties than the multiquadric and Gaussian RBFs. Moreover, numerical investigations that do not use evolutionary algorithms suggest that cubic RBFs are better than thinplate and multiquadric RBFs [9].

IV. An Enhanced Evolutionary Algorithm for Costly Black Box Optimization

A. General Description

The main idea behind this approach to costly black box optimization is to reduce the number of costly function evaluations in each generation of an evolutionary algorithm by estimating the function values of the offspring using local response surface approximations. Using the estimated function values of the offspring, we select a subset of the offspring where the costly function will be evaluated. Given a limit on the function evaluations of an evolutionary optimization algorithm, the reduction of costly evaluations in each generation will allow us to run the evolutionary algorithm for more generations than is possible with the standard implementation.

We now describe the details of the proposed approach. Assume that we are given an evolutionary algorithm (EA) and that we wish to optimize a continuous function f defined on a box $\mathcal{D} \subseteq R^d$. First, in order to do response surface approximation, we need to know the values of f at some initial sample of points in $\mathcal{D}$. An excellent way to do this is to select a particular space-filling experimental design, evaluate the costly function at each of the design points, and store the data so that it can be used for fitting response surface models. In the next section, we describe the space-filling experimental design that was used in this investigation. Second, since we already know the function values at the experimental design points, we select a subset of the design points to become the initial parent population. We then proceed to generate offspring as we would in an ordinary implementation of the EA.

Typically, in each generation of an EA, we generate a fixed number of offspring (usually a large number) via mutation and/or recombination. In the standard implementation of an EA, we would evaluate the costly objective function $f(x)$ at each of these offspring. In the proposed procedure, we first estimate the function value at each of these offspring using local approximation as described earlier in Section III. Based on the estimates of the function values of the offspring, we select offspring that have the lowest function value and perform the costly function evaluation on them. Once we have performed the costly evaluation, we proceed with the evolutionary algorithm as though we only generated the offspring for which the costly function was evaluated. That is, the selection process for the parents of the next generation will then be restricted to the offspring for which the actual function values have been calculated plus the set $Q(t)$ (in Figure 1) in the case of an elitist algorithm.

Finally, note that in a standard implementation of an evolutionary optimization algorithm, only the

function values of the current population are normally retained. Hence, a lot of good information on the function values of other points computed in prior generations are discarded. In contrast, in the proposed enhanced EA, all information on the previously evaluated points are stored and can be used for estimating the function values of new offspring. In this sense, the proposed enhanced EA is less wasteful than the standard implementation of the same EA.

Figure 2 shows a pseudo-code of an enhanced evolutionary optimization algorithm for costly black box functions. In the context of enhancing a (μ, λ)-ES, the resulting algorithms will be called (μ, λ, ν)-ESQR for quadratic regression and (μ, λ, ν)-ESRBF for radial basis function interpolation. In this notation, μ is still the number of parents, λ is still the number of offspring generated in each generation, and ν is the fixed number of offspring out of λ that will be selected for costly function evaluation.

B. Space-Filling Experimental Designs

There are several types of experimental designs that we can use to get the initial points for the response surface approximations. Koehler and Owen [32] describe various ways of choosing the experimental design points for computer experiments. In this investigation, we concentrate only on Latin hypercube designs.

One desirable characteristic of Latin hypercube designs is that the user can specify the number of design points. In addition, if we project these design points onto any single dimension, then the result is a regular grid in one dimension. Latin hypercube designs were proposed by McKay et al. [53]. A procedure for obtaining a Latin hypercube design (LHD) of size m is shown below. In the algorithm below, assume that the domain $\mathcal{D}$ is given by (1).

Construction of a Random LHD

(1). For each $j = 1, \ldots, d$, partition the interval $[a_j, b_j]$ into m sub-intervals of equal length and let $c_j^{(i)}$ denote the midpoint of the i^{th} sub-interval of $[a_j, b_j]$.

(2). For each $j = 1, \ldots, d$, randomly select a permutation of $1, \ldots, m$ and denote it by π_j.

(3). Now for each $i = 1, \ldots, m$, the i^{th} Latin hypercube design point is given by

$$(c_1^{(\pi_1(i))}, c_2^{(\pi_2(i))}, \ldots, c_d^{(\pi_d(i))}).$$

Observe that if we fix the box domain of f, then an LHD is completely determined by the d permutations selected in step 3. Hence, an LHD of size m for R^d may also be defined as an $m \times d$ array whose columns are permutations of $1, \ldots, m$. In fact, in the computational experiments, we generated Latin hypercube designs only for domains of the form $[0, 1]^d$ since these designs can be rescaled and used for any box domain.

Another thing to note from the above LHD description is that the procedure yields a random LHD. However, in practice, some randomly generated LHDs may have poor estimation and prediction properties. Hence, it is not enough to simply pick a particular LHD. We need to somehow impose some optimality conditions on the LHDs. In the literature, there are several papers that deal with optimal Latin hypercubes and they also discuss ways of creating optimal LHDs ([33], [54], [55]). However, finding an optimal LHD can

be time consuming. Now a recent paper by Ye et al. [34] provides a compromise between computing effort and design optimality. They proposed using symmetric LHDs instead of ordinary LHDs. A *symmetric Latin hypercube design* (SLHD) of size m is an $m \times d$ LHD (in permutation notation as described above) with the following property: if $(a_1, \ldots, a_d)$ is one of the rows, then the vector $(m+1-a_1, \ldots, m+1-a_d)$ must be another row in the design matrix. That is, an SLHD is simply an LHD whose design points are symmetric about the center of the box domain. The authors showed that SLHDs have some advantages over the regular LHDs with respect to criteria such as entropy and minimum intersite distance. Hence, for this investigation, we have used a randomly generated SLHD to get an initial sample of points where the costly function will be evaluated. A procedure for constructing randomly generated SLHDs is given below. Note that we only need to replace step 2 of the procedure for constructing an LHD.

Construction of a Random SLHD

(1). Initialize an array M of size $m \times d$.

(2). If m is odd, set $M(\frac{m+1}{2}, j) = \frac{m+1}{2}$ for $j = 1, \ldots, d$.

(3). Define $k := \lceil \frac{m-1}{2} \rceil$.

(4). For each $j = 1, \ldots, d$, randomly select a permutation of $1, \ldots, k$ and denote it by ψ_j.

(5). For each pair (i, j), where $i = 1, \ldots, k$ and $j = 1, \ldots, d$,

(a). Generate a uniform random number $\omega_{i,j} \in [0, 1]$.

(b). If $\omega_{i,j} \leq 1/2$, and set $M(i, j) = \psi_j(i)$ and $M(m+1-i, j) = m+1-\psi_j(i)$. Otherwise, set $M(i, j) = m+1-\psi_j(i)$ and $M(m+1-i, j) = \psi_j(i)$.

(6). For each $j = 1, \ldots, d$, let π_j be the j^{th} column of M and continue with Step 3 of the above LHD construction.

V. Computational Experiments

A. Test Functions

Computational experiments were performed on some benchmark test functions for global optimization in order to compare the performance of ES with its enhanced versions. The test functions include the classical Dixon-Szegö test functions for global optimization [36] and the 10-dimensional Rastrigin and Ackley test functions (which we refer to as Rastrigin10 and Ackley10, respectively). These functions are not really costly to evaluate but their shapes are complex and multimodal, and hence, the relative performance of algorithms on these test functions is expected to mimic performance on costly functions. Table I shows the characteristics of the Dixon-Szegö test functions. The actual functional expressions for the Dixon-Szegö test functions can be found in [36]. In this paper, we will use the following versions of the d-dimensional Rastrigin and Ackley test functions:

$$\text{(Rastrigin)} \qquad f(x) = \sum_{i=1}^{d} (x_i^2 - \cos(2\pi x_i)), \quad x_i \in [-2, 2]$$

TABLE I

THE DIXON-SZEGÖ TEST FUNCTIONS [36].

Test Function	*Dim*	*Domain*	*Number of Local Min*	*Number of Global Min*	*Approx Global Min Value*
Branin	2	$[-5,10] \times [0,15]$	3	3	0.398
Goldstein-Price	2	$[-2,2]^2$	4	1	3
Hartman3	3	$[0,1]^3$	4	1	-3.86
Shekel5	4	$[0,10]^4$	5	1	-10.1532
Shekel7	4	$[0,10]^4$	7	1	-10.4029
Shekel10	4	$[0,10]^4$	10	1	-10.5364
Hartman6	6	$[0,1]^6$	4	1	-3.32

$$\text{(Ackley)} \qquad f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right), \qquad x_i \in [-2,2]$$

The d-dimensional Rastrigin and Ackley test functions are both highly multimodal with a unique global minimum point at $(0,\ldots,0)$. The global minimum values for the d-dimensional Rastrigin and Ackley test functions are $-d$ and $-20-e$, respectively.

B. Specific Algorithms Used for the Experiments

We shall compare the performance of a (μ,λ)-ES, (μ,λ,ν)-ESQR, (μ,λ,ν)-ESRBF, and (μ,ν)-ES on the test functions for selected values of μ,λ,ν such that $\mu \leq \nu \leq \lambda$. In particular, we use $\mu = 8$, $\lambda = 50$, and $\nu = 20$ for the Branin, Goldstein-Price, and Hartman3 test functions; we use $\mu = 16$, $\lambda = 100$, and $\nu = 40$ for the Shekel and Hartman6 test functions; and finally, we use $\mu = 32$, $\lambda = 200$, and $\nu = 80$ for the Rastrigin10 and Ackley10 test functions. Note that the values of μ and λ are in accordance with Schwefel's recommendation that $\mu/\lambda \approx 1/7$ [52]. The (μ,λ,ν)-ESQR and the (μ,λ,ν)-ESRBF are modified versions of an ES with μ parents that perform exactly ν function evaluations in each generation. Since these algorithms somehow behave like a (μ,ν)-ES, it is also necessary to compare these algorithms with the (μ,ν)-ES.

C. Number of Experimental Design Points

Recall that in order to implement the enhanced versions of an ES, we need to allocate some function evaluations for the experimental design. For quadratic regression, we need a minimum of $(d+1)(d+2)/2$ points where the function values are known. For RBF interpolation, there is really no requirement on the number of points. In general, determining the number of points to allocate for the experimental design is not easy. If we allocate too few for the experimental design, then the approximations will be poor. On the other hand, if we allocate too many then we would be wasting function evaluations that are better spent in the actual optimization. The choice of the number of experimental design points should be based on

whatever knowledge is available on the underlying response surface (i.e. does it have many bumps or is it quite smooth) as well as the maximum number of function evaluations allowed. In the implementation of the ESQR and ESRBF, we standardized the approach for all test functions by fixing the number of experimental design points to 4 times the minimum number required for quadratic regression. That is, for each trial of an ESQR or ESRBF, we randomly generated a symmetric Latin hypercube design of size $2(d+1)(d+2)$. Note that the ESRBF does not really need this number of experimental design points to get started but the number was set the same for both the ESQR and ESRBF for comparison purposes.

D. Comparison of the Algorithms

We now describe the actual experiments that were performed to compare the different algorithms on the test functions. For each of the test functions, each of the two ES algorithms were run 100 times, each time using a different randomly generated set of initial parent solutions. Each of the enhanced ES algorithms (ESQR and ESRBF) were also run 100 times, each time using a different randomly generated SLHD. Moreover, for comparison purposes, we made the set of experimental design points identical for ESQR and ESRBF in any given run.

The performance of the algorithms were compared based on the mean of the best values obtained in the different runs. The results are summarized in Figures 3-11. For the moment, ignore the error bars on the plots. To highlight the differences in performance, the plots were started at m function evaluations, where $m = 2(d+1)(d+2)$ is the number of experimental design points used for ESQR and ESRBF.

From the plots in Figures 3-9, we can see that the (8,50,20)-ESQR outperforms the (8,50)-ES and the (8,20)-ES in terms of the mean best value on the Branin, Goldstein-Price and Hartman3 test functions. Similarly, the (16,100,40)-ESQR outperforms the (16,100)-ES and the (16,40)-ES on the Shekel5, Shekel7, Shekel10 and Hartman6 test functions. The same observations can be said about the (8,50,20)-ESRBF and the (16,100,40)-ESRBF. In fact, the performance of the ESRBF appears to be only slightly better than that of the ESQR on the Dixon-Szegö test functions and this difference in performance is not statistically significant as will be seen in section E.

From the plots in Figures 10 and 11, the ESRBF algorithm is better than the two ES algorithms and also better than the ESQR algorithm on the harder 10-dimensional test functions. Moreover, for these test functions, only the ESRBF scheme was helpful in improving the performance of a $(32, 200)$-ES. These results suggest that RBF models are much better than quadratic models at capturing the complexities of highly multimodal test functions.

E. Statistical Analysis of the Results

We performed single-factor analysis of variance (ANOVA) on the results at several uniformly spaced evaluation points for each test function. The factor under consideration is the algorithm and it has four levels corresponding to the (μ, λ)-ES, (μ, λ, ν)-ESQR, (μ, λ, ν)-ESRBF, and (μ, ν)-ES. The goal of the single-factor ANOVA is to determine whether the observed differences in the means of the best values of the four

algorithms (at a fixed evaluation point and for a given test function) are statistically significant. A brief description of the single-factor ANOVA model is in the appendix. The results of ANOVA indicate that the mean best values are significantly different at several evaluation points in all test functions.

After we have established that the means of the best values for the four algorithms are different for some test function at a certain evaluation point, the next step is to determine which algorithm is significantly better than another for the same test function and the same evaluation point. This is can be accomplished by performing multiple comparison procedures which are normally integrated with the ANOVA procedure ([56], [57]).

There are essentially two ways of performing multiple comparisons of the factor level means. In the traditional approach, the differences between every pair of factor level means are simultaneously tested and compared to zero. Here, simultaneous testing means that the level of significance (or the probability of a Type I error) is controlled for a family of hypotheses instead of for a single hypothesis. Differences that are significantly different from zero indicate that one of the algorithms involved in the difference is significantly better than the other. The alternative approach is to construct simultaneous confidence intervals ([56], [58]) for the means of the best values for each algorithm. Significant differences between the means of the best values exist only when the confidence intervals do not overlap. We have used the latter approach since it is more convenient for presentation purposes. In the appendix, we describe how these simultaneous confidence intervals are constructed.

To determine which algorithms resulted in significantly lower mean best values, we constructed simultaneous confidence intervals of the form (31) (in the appendix) for the mean best values of the different algorithms at several evaluation points. These simultaneous confidence intervals were represented as error bars in the plots on Figures 3-11. The level of significance for the confidence intervals was set at $\alpha = 0.05$.

For the Dixon-Szegö test functions, the simultaneous confidence intervals indicate that the mean best value of the (μ, λ, ν)-ESQR and (μ, λ, ν)-ESRBF algorithms are in fact significantly better than that of the (μ, λ)-ES and the (μ, ν)-ES algorithms for all of the evaluation points considered except at the beginning or at the later evaluations points where most of the trials (of the different algorithms) have converged. Moreover, although the (μ, λ, ν)-ESRBF algorithm appears to be slighly better than the (μ, λ, ν)-ESQR algorithm on the Dixon-Szegö test functions (see the plots in Figures 3-9), the difference in performance between the two algorithms is generally not significant.

For the Rastrigin10 and Ackley10 test functions, the simultaneous confidence intervals indicate that the (μ, λ, ν)-ESRBF algorithm was significantly better than all the other algorithms at all evaluation points considered. Moreover, for these test functions, the (μ, λ, ν)-ESQR algorithm was no longer better the the (μ, λ)-ES and the (μ, ν)-ES algorithms. The results on these two highly multimodal test functions indicate that the RBF approach is to be preferred over quadratic regression when there is reason to believe that the function to be optimized is highly multimodal.

VI. Summary and Conclusions

In this paper, we have shown that the performance of an evolutionary algorithm can be substantially improved with the use of a space-filling experimental design and local response surface approximation via k-nearest neighbors, where $k = (d+1)(d+2)/2$ and d is the dimension of the problem. In the proposed scheme, the objective function value of each offspring is first estimated by fitting a response surface model using the nearest $(d+1)(d+2)/2$ previously evaluated points, and then a subset of the offspring solutions are selected for costly function evaluation based on the estimates of their objective function value.

We have compared four algorithms: a (μ, λ)-ES, a (μ, λ, ν)-ESQR (an ES which uses local quadratic approximation), a (μ, λ, ν)-ESRBF (an ES which uses local augmented cubic RBF interpolation), and a (μ, ν)-ES. Both local response surface algorithms use a symmetric Latin hypercube design to select points for initial function evaluations. The performance of the four algorithms were compared on nine benchmark test functions for global optimization: the seven Dixon-Szegö test functions, and the 10-dimensional Rastrigin and Ackley test functions. We performed multiple trials for each algorithm on each test problem, analyzed the results with ANOVA, and compared the performance of the different algorithms using simultaneous confidence intervals.

The statistical analysis indicates that the ES plus local augmented cubic RBF interpolation is significantly better than the conventional ES algorithms on all test functions. The ES plus local quadratic approximation is significantly better than the ES algorithms without local approximation on the Dixon-Szegö test functions. However, for the harder 10-dimensional test functions, the ES plus local quadratic approximation was not much better, and sometimes worse, than the (μ, λ)-ES. Moreover, the RBF approach was consistently better than the quadratic approximation approach on all test functions although the difference in performance is statistically significant only on the harder test functions (Rastrigin10 and Ackley10). This is the first evidence of the superiority of an RBF approach with an evolutionary algorithm since the only prior study comparing RBF networks and quadratic regression with an evolutionary algorithm used a Gaussian type of RBF, which was shown to be inferior to a quadratic approximation [30].

The results of the computational experiments, which were subjected to rigorous statistical analysis, suggest that the approach that uses space-filling experimental designs together with local response surface approximations via k-nearest neighbors has potential for success in enhancing evolutionary algorithms for computationally expensive real-world problems. Moreover, the RBF approach for local response surface approximation appears to be more promising than the quadratic approximation approach on more difficult higher-dimensional problems.

Acknowledgements

We would like to thank the Intelligent Information Systems Institute (IISI) directed by Dr. Carla Gomes for providing GRA funding for Rommel Regis (AFOSR, grant F49620-01-1-0076). We would also like to thank Prof. David Ruppert and Prof. Bruce Turnbull of the School of Operations Research and Industrial Engineering, Prof. Rich Caruana of the Department of Computer Science, and Dr. Sophonie Nshinyabakobeje of

the Department of Biological Statistics and Computational Biology for their technical input and comments.

References

[1] G.E.P. Box and N.R. Draper, *Empirical Model-Building and Response Surfaces*, John Wiley & Sons, Inc., New York, 1987.

[2] A.I. Khuri and J.A. Cornell, *Response Surfaces*, Marcel Dekker, Inc., New York, 1987.

[3] R.H. Myers and D.C. Montgomery, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, John Wiley & Sons, Inc., New York, 1995

[4] D. Jones, "A Taxonomy of Global Optimization Methods Based on Response Surfaces," *Journal of Global Optimization*, vol. 21, no. 4, pp. 345-383, 2001.

[5] D.R. Jones, M. Schonlau and W.J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455-492, 1998.

[6] J. Sacks, W.J. Welch, T.J. Mitchell and H.P. Wynn, "Design and Analysis of Computer Experiments," *Statistical Science*, vol. 4, no. 4, pp. 409-435, 1989.

[7] T.W. Simpson, T.M. Mauery, J.J. Korte and F. Mistree, Comparison of Response Surface and Kriging Models for Multidisciplinary Design Optimization, *Proc. 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MO, vol. 1, pp. 381-391, 1998.

[8] M. Björkman and K. Holmström, "Global Optimization of Costly Nonconvex Functions Using Radial Basis Functions," *Optimization Engineering*, vol. 1, no. 4, pp. 373-397, 2000.

[9] H.-M. Gutmann, "Radial Basis Function Methods for Global Optimization," PhD Thesis, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Sept. 2001.

[10] H.-M. Gutmann, "A Radial Basis Function Method for Global Optimization," *Journal of Global Optimization*, vol. 19, no. 3, pp. 201-227, 2001.

[11] T. Ishikawa and M. Matsunami, "An Optimization Method Based on Radial Basis Functions," *IEEE Transactions on Magnetics*, vol. 33, no. 2, pp. 1868-1871, 1997.

[12] T. Ishikawa, Y. Tsukui and M. Matsunami, "A Combined Method for the Global Optimization Using Radial Basis Function and Deterministic Approach," *IEEE Transactions on Magnetics*, vol. 35, no. 3, pp. 1730-1733, 1999.

[13] M.J.D. Powell, The theory of Radial Basis Function Approximation in 1990, in *Advances in Numerical Analysis, Volume 2: Wavelets, Subdivision Algorithms and Radial Basis Functions*, W. Light, Ed. Oxford University Press, pp. 105-210, 1992.

[14] M.J.D. Powell, Recent research at Cambridge on radial basis functions, in *New Developments in Approximation Theory, International Series of Numerical Mathematics, Vol. 132*, M. Muller, M. Buhmann, D. Mache and M. Felten, Eds., Birkhauser Verlag, Basel, pp. 215-232, 1999.

[15] W. Chen, and S. Varadarajan, Integration of Design of Experiments and Artificial Neural Networks for Achieving Affordable Concurrent Design, 38^{th} *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference* and *AIAA/ASME/AHS Adaptive Structures Forum*, AIAA-97-1230, Kissimmee, FL, 1997.

[16] W. Liu and S.M. Batill, Gradient-Enhanced Neural Network Response Surface Approximations, AIAA Paper 2000-4923, *AIAA Multidisciplinary Analysis and Optimization Conference and Exhibit*, Long Beach, California, 2000.

[17] D. Padmanabhan and S.M. Batill, An Iterative Concurrent Subspace Robust Design Framework, AIAA Paper 2000-4841, *AIAA Multidisciplinary Analysis and Optimization Conference and Exhibit*, Long Beach, California, 2000.

[18] R.S. Sellar and S.M. Batill, Concurrent Subspace Optimization Using Gradient-Enhanced Neural Network Approximations, AIAA Paper 96-4019, 6^{th} *Annual AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, Washington, 1996.

[19] R.S. Sellar, S.M. Batill, and J.E. Renaud, Response Surface Based, Concurrent Subspace Optimization for Multidisciplinary System Design, AIAA Paper 96-0714, *AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 1996.

[20] A. Ratle, "Accelerating the convergence of evolutionary algorithms by fitness landscape approximation," in *Parallel Problem Solving from Nature - PPSN V, Springer Verlag, Lecture Notes in Computer Science*, T. Bäck, A.E. Eiben, M. Schoenauer, and H.-P. Schwefel, Eds., Berlin: Springer, 1998, pp. 87-96.

[21] A. Ratle, "Optimal sampling strategies for learning a fitness model," in *Proc. of the 1999 Congress on Evolutionary Computation*, vol. 3, pp. 2078-2085, Piscataway, NJ, 1999, IEEE Press.

[22] M. El-Beltagy, P. Nair, and A. Keane, "Metamodeling Techniques for Evolutionary Optimization of Computationally Expensive Problems: Promises and Limitations," in *Proc. of the Genetic and Evolutionary Computation Conference, vol. 1*, W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, and R.E. Smith, Eds., San Francisco, California: Morgan Kaufmann, 1999, pp. 196-203.

[23] M.A. El-Beltagy and A.J. Keane, "Evolutionary Optimization for Computationally Expensive Problems using Gaussian Processes," in *Proc. of the Int. Conf. on Artificial Intelligence IC-AI '2001*, Volume II, Hamid Arabnia, Ed., CSREA Press, pp. 708-714, Las Vegas, Nevada, 2001.

[24] Y. Jin, M. Olhofer, and B. Sendhoff, "On Evolutionary Optimization with Approximate Fitness Functions,", in *Proc. of the Genetic and Evolutionary Computation Conference*, pp. 786-792, Las Vegas, 2000.

[25] Y. Jin, M. Olhofer, and B. Sendhoff, "Managing Approximate Models in Evolutionary Aerodynamic Design Optimization," in *Proc. of the IEEE Congress on Evolutionary Computation*, vol. 1, pp. 592-599, Seoul, Korea, May 2001.

[26] Y. Jin, M. Olhofer, and B. Sendhoff, "A Framework for Evolutionary Optimization with Approximate Fitness Functions," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 481-494, 2002.

[27] K. Rasheed, "An Incremental-Approximate-Clustering Approach for Developing Dynamic Reduced Models for Design Optimization," in *Proc. of the Congress on Evolutionary Computation* (CEC), 2000.

[28] K. Rasheed, X. Ni, and, S. Vattam, "Comparison of Methods for Developing Dynamic Reduced Models for Design Optimization," in *Proc. of the Congress on Evolutionary Computation* (CEC), 2002.

[29] Y. Jin, "Fitness Approximation in Evolutionary Computation - A Survey," in *Proc. of the Genetic and Evolutionary Computation Conference*, pp. 1105-1112, New York, July 2002.

[30] K. Rasheed, S. Vattam, and X. Ni, "Comparison of Methods for Using Reduced Models to Speed Up Design Optimization," in *Proc. of the Genetic and Evolutionary Computation Conference* (GECCO), 2002.

[31] H.-M. Gutmann, "On the Semi-Norm of Radial Basis Function Interpolants," Technical Report DAMTP 2000/NA04, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, England, 2000.

[32] J.R. Koehler and A.B. Owen, "Computer Experiments," in *Handbook of Statistics, 13: Design and Analysis of Computer Experiments*, S. Ghosh and C.R. Rao, Eds., North-Holland, 1996, pp. 261-308.

[33] M.D. Morris and T.J. Mitchell, "Exploratory Designs for Computational Experiments," *Journal of Statistical Planning and Inference*, vol. 43, pp. 381-402, 1995.

[34] K.Q. Ye, W. Li and A. Sudjianto, "Algorithmic Construction of Orthogonal Symmetric Latin Hypercube Designs," *Journal of Statistical Planning and Inference*, vol. 90, 2000.

[35] H.-P. Schwefel, *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, vol. 26 of *Interdisciplinary Systems Research*, Birkhäuser, Basel, 1977.

[36] L.C.W. Dixon, and G. Szegö, The Global Optimization Problem: An Introduction, in *Towards Global Optimization 2*, L.C.W. Dixon and G. Szegö, Eds., North-Holland, Amsterdam, pp. 1-15, 1978.

[37] L.A. Rastrigin, *Systems of Extremal Control*, Nauka, Moscow, 1974. (in Russian)

[38] D.H. Ackley, *A Connectionist Machine for Genetic Hillclimbing*, Kluwer, Boston, 1987.

[39] T. Bäck and H.-P. Schwefel, Evolutionary Computation: An Overview, *Proc. of the 1996 IEEE Int. Conf. on Evolutionary Computation*, 1996.

[40] T. Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York, 1996.

[41] D.B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, Piscataway, New Jersey, 1995.

[42] J.H. Holland, "Outline for a Logical Theory of Adaptive Systems," *Journal of the Association of Computing Machinery*, vol. 3, pp. 297-314, 1962.

[43] J.H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI, 1975.

[44] I. Rechenberg, "Cybernetic Solution Path of an Experimental Problem," Royal Aircraft Establishment, Library translation No. 1122, Farnborough, Hants., UK, August 1965.

[45] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Holzboog, Stuttgart, 1973.

[46] I. Rechenberg, *Evolutionsstragie '94*, volume 1 of *Werkstatt Bionik und Evolutionstechnik*, Frommann-Holzboog, Stuttgart, 1994.

[47] H.-P. Schwefel, *Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik*, Diplomarbeit, Technische Universität Berlin, 1965.

[48] H.-P. Schwefel, *Numerical Optimization of Computer Experiments*, Wiley, Chichester, 1981.

[49] H.-P. Schwefel, *Evolution and Optimum Seeking*, Sixth-Generation Computer Technology Series, Wiley, New York, 1995.

[50] L.J. Fogel, "Toward Inductive Inference Automata," in *Proceedings of the International Federation for Information Processing Congress*, pp. 395-399, Munich, 1962.

[51] L.J. Fogel, A.J. Owens, and M.J. Walsh, *Artificial Intelligence through Simulated Evolution*, Wiley, New York, 1966.

[52] H.-P. Schwefel, "Collective Phenomena in Evolutionary Systems," in *Preprints of the 31st Annual Meeting of the International Society for General System Research, Budapest*, vol. 2, pp. 1025-1033, June 1987.

[53] M. McKay, R. Beckman and W. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, pp. 239-246, 1979.

[54] J.-S. Park, "Optimal Latin-hypercube Designs for Computer Experiments," *Journal of Statistical Planning and Inference*, vol. 39, pp. 95-111, 1994.

[55] W. Li, "Optimal Designs Using CP Algorithms," in *Proc. for the 2nd World Conference of the International Association for Statistical Computing*, pp. 130-139, 1997.

[56] R.G. Miller, Jr., *Simultaneous Statistical Inference*, Springer-Verlag, New York, 1981.

[57] J. Neter, W. Wasserman, and M.H. Kutner, *Applied Linear Statistical Models*, Richard D. Irwin, Inc., Illinois, 1985.

[58] P.J. Bickel and K.A. Doksum, *Mathematical Statistics*, Prentice-Hall, Inc., New Jersey, 1977.

Appendix

I. Single-Factor Analysis of Variance

Detailed material on analysis of variance procedures can be found in [56], [57], [58]. Below is a short summary of the material in [57].

The *fixed effects single-factor ANOVA model* with r factor levels (or treatments) and n observations at each factor level may be described as follows:

$$Y_{ij} = \mu_i + \epsilon_{ij}, \qquad i = 1, \ldots, r; j = 1, \ldots, n \tag{19}$$

where Y_{ij} is the random variable representing the j^{th} observation for the i^{th} factor level, μ_i is the parameter associated with the i^{th} factor level, and ϵ_{ij} are independent $N(0, \sigma^2)$. It is easy to check that this model is equivalent to the following:

$$Y_{ij} \text{ are independent } N(\mu_i, \sigma^2) \tag{20}$$

In the context of our investigation, the factor under consideration is the algorithm. It has 4 levels corresponding to the two conventional ES algorithms, ESQR and ESRBF. Moreover, Y_{ij} is the best function value encountered by the i^{th} algorithm during the j^{th} trial.

Applying a fixed effects single-factor ANOVA on a data set essentially means fitting a model of the form (20) using the method of least squares. The least squares estimator for μ_i is given by the i^{th} factor level sample mean which is given by

$$\hat{\mu}_i = \bar{Y}_{i\cdot} = \frac{\sum_{j=1}^{n} Y_{ij}}{n} \tag{21}$$

Hence, the fitted value for the observation Y_{ij} is simply $\hat{Y}_{ij} = \bar{Y}_{i\cdot}$. Moreover, the *error mean square*, which is defined by

$$\text{MSE} = \frac{\sum_{i=1}^{r}\sum_{j=1}^{n}(Y_{ij} - \bar{Y}_{i\cdot})^2}{r(n-1)} \tag{22}$$

is an unbiased estimator of σ^2.

When fitting any model to a set of data, it is important to check whether the assumptions of the model are satisfied. In the case of the fixed effects single-factor ANOVA model, this can be accomplished by examining the residuals

$$e_{ij} = Y_{ij} - \hat{Y}_{ij} = Y_{ij} - \bar{Y}_{i\cdot} \tag{23}$$

Ideally, the e_{ij}'s should behave like independent and normally distributed random variables with constant variance. If there are significant departures from this assumption, then we either take corrective measures such as transformation of the data or we modify the model.

Assume that we have fitted an ANOVA model to data set, we have examined its residuals and found that the model assumptions are satisfied (i.e. the model is appropriate for the data). Now we can proceed with the analysis of the data. In a typical ANOVA implementation, one is generally interested in determining whether or not the factor level means μ_i are equal. This can be accomplished by testing the following hypotheses:

$$\begin{aligned} &H_O\text{: } \mu_1 = \mu_2 = \ldots = \mu_r \\ &H_A\text{: not all } \mu_i \text{ are equal} \end{aligned} \tag{24}$$

The test statistic that is used for choosing between the above alternatives is given by

$$F^* = \frac{\text{MSTR}}{\text{MSE}} \tag{25}$$

where

$$\text{MSTR} = \frac{\sum_{i=1}^{r} n(\bar{Y}_{i\cdot} - \bar{Y}_{\cdot\cdot})^2}{r-1} \text{ and } \bar{Y}_{\cdot\cdot} = \frac{\sum_{i=1}^{r}\sum_{j=1}^{r} Y_{ij}}{rn} \tag{26}$$

Now if we set the level of significance at α (the probability of Type I error), then the decision rule (also known as the F test) is given by

$$\begin{cases} \text{do not reject } H_O & \text{if } F^* \le F(1-\alpha; r-1, r(n-1)) \\ \text{reject } H_O & \text{if } F^* > F(1-\alpha; r-1, r(n-1)) \end{cases} \tag{27}$$

where $F(1-\alpha; r-1, r(n-1))$ is the $(1-\alpha)100$ percentile of the F distribution with $r-1$ and $r(n-1)$ degrees of freedom.

II. Simultaneous Confidence Intervals

The F test for determining whether or not the factor level means μ_i differ is just a preliminary test. If the F test does not lead to a rejection of the null hypothesis H_O, then we conclude that there is no relation

between the factor under consideration and the dependent variable and we are done. On the other hand, if the F test resulted in the rejection of the null hypothesis, then we know that not all factor level means are equal. An obvious next step is to determine which factor level means differ significantly from other factor level means. One way to accomplish this is by means of the Tukey method of multiple comparisons ([58], [56], [57]) where the goal is to find confidence intervals I_{kl}, $1 \leq k < l \leq r$, such that

$$Pr[\mu_l - \mu_k \in I_{kl}, 1 \leq k < l \leq r] \geq 1 - \alpha. \tag{28}$$

This can be done by considering

$$I_{kl} = (\bar{Y}_{l\cdot} - \bar{Y}_{k\cdot}) \pm d(1-\alpha; r, (n-1)r)\sqrt{\frac{\text{MSE}}{n}} \tag{29}$$

where $d(1-\alpha; r, (n-1)r)$ is the $(1-\alpha)100$ percentile of the *studentized range distribution* with parameters r and $(n-1)r$. Values for this distribution can be found in Miller [56]. Now if a confidence interval I_{kl} contains zero, then we conclude that the difference between μ_k and μ_l is not statistically significant. Otherwise, the difference is statistically significant.

Presenting confidence intervals for every pair of factor level means may be cumbersome if there are several factor levels. An alternative would be to find confidence intervals I_i, $i = 1, \ldots, r$ such that

$$Pr[\mu_i \in I_i, i = 1, \ldots, r] \geq 1 - \alpha \tag{30}$$

This is equivalent to finding simultaneous confidence intervals about the factor level means. This can be done by considering

$$I_i = \bar{Y}_{i\cdot} \pm q(1-\alpha; r, (n-1)r)\sqrt{\frac{\text{MSE}}{n}} \tag{31}$$

where $q(1-\alpha; r, (n-1)r)$ is the $(1-\alpha)100$ percentile of the *studentized maximum modulus distribution* with parameters r and $(n-1)r$. Values for this distribution can also be found in Miller [56].

Rommel G. Regis is a Ph.D. candidate in Operations Research at Cornell University. He received a Master's degree in Mathematics at the University of Florida in 1998 and a Master's degree in Operations Research at Cornell in 2002. He also has an extensive background in Computer Science and Statistics. His current research interests include numerical optimization and machine learning.

Christine Shoemaker (M '90) is the Joseph P. Ripley Professor of Engineering at Cornell University. She received her Ph.D. in mathematics with a specialty in dynamic programming and optimal control theory under the supervision of Richard Bellman in Electrical Engineering at the University of Southern California. She received a Humboldt Research Prize in 2001 and the Hinds Award from ASCE in 1999. In addition to her research on serial and parallel optimization algorithms, she also works on optimization applications to problems in environmental engineering, which include optimization of large systems of partial differential equations. She teaches a Computer Science graduate course at Cornell on heuristic optimization methods with applications in computer science and engineering.